AN ABSTRACT OF THE THESIS OF

Burçin Aktan for the degree of Master of Science in

Electrical And Computer Engineering presented on February 9, 1996.

Title: Distance Learning Applied to Control Engineering Education.

Redacted for privacy

Abstract approved: _____

Molly H. Shor

At the College of Engineering, Oregon State University, we have successfully developed and demonstrated a distance learning application utilizing emerging communications technologies in a new way to allow control engineering teaching laboratories to be accessed remotely. Second Best to Being There (SBBT) provides the remote user with all the capabilities of a local lab user through the laboratory environment control interface. This allows the remote student to develop, compile, debug, and run controllers in real-time on the experiments in our laboratory. The students are provided with live video and audio tools, giving them the sense of presence in the laboratory, and collaboration tools, facilitating interactions among users. SBBT has been implemented by effectively interfacing the student and the laboratory to the Internet.

Distance Learning Applied to Control Engineering Education

by

Burçin Aktan

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed February 9, 1996
Commencement June 1996

Master of Science thesis of Burçin Aktan presented on February 9, 1996

APPROVED:

Redacted for privacy

Major Professor, representing Electrical And Computer Engineering

Redacted for privacy

Chair of the Department of Electrical and Computer Engineering

Redacted for privacy

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for privacy

Burçin Aktan, Author

ACKNOWLEDGMENT

I would like to thank my sister, Gülçin Aktan, for her invaluable help in the demonstrations, my thesis advisor Molly H. Shor, for her guidance and Carisa A. Bohus for being a great project partner. I also wish to acknowledge Stephen Wilcox for making the Motor Control Interface a reality.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

PREFACE

Second Best to Being There is a joint project between the Computer Science and Electrical and Computer Engineering Departments of Oregon State University. Carisa Bohus from CS handled most of the CS issues and I, Burçin Aktan, have been responsible mainly for the ECE and laboratory education issues. Most of the paradigm had been outlined by Carisa Bohus by interviewing numerous people, before I joined the project. During the project development, we complemented each other and made sure each piece fit together with each other's understanding and expertise regarding the project. While Carisa developed the client/server software to connect the server to the Internet, I worked on the following issues:

- Identifying and analyzing major Distance Learning issues.
- Analyzing and outlining laboratory education and making sure the ECE view of lab teaching and the application of individual paradigm components fit with the paradigm.
- Development of the Graphical User Interface(GUI) and interfacing of it with the client software.
- Workstation to PC communications and interfacing it with the server software. This includes modifying portions of the client/server code to interface better with the communication routines I have written.
- Designing the demonstration project; designing the experiment and programming the controller for the robot.
- Analyzing the user and application level issues including the optimal usage of multimedia tools related to the project.

SBBT has produced three publications. The *IFAC World Congress* paper [1] and the technical report [2] with authors Carisa Bohus, Burçin Aktan, Lawrence Crowl, and Molly Shor, outline the first findings and the major issues involved in the application development. The main focus of these is the distance laboratory paradigm and application. The latest work on the topic is the manuscript accepted for publication by *IEEE Transactions on Education* [3]. This work was adapted from the conference paper and the technical report. The main focus and additions in this work, authored by Burçin Aktan, Carisa Bohus, Lawrence Crowl, and Molly Shor, are distance learning issues related to engineering laboratories, comparisons to the latest related research, and the networking issues.

This thesis summarizes the author's contributions to the project. It includes detailed descriptions of the user interface, hardware/software design issues related to the in-lab communications, issues concerning the components of our multimedia application, and most important of all, recent experience using the overall application.

DISTANCE LEARNING APPLIED TO CONTROL ENGINEERING LABORATORIES

1. INTRODUCTION

Second Best to Being There is a distance learning application that allows remotely-located control engineering students to access laboratory resources at Oregon State University. SBBT uses the Internet as the medium of main connectivity. The students are able to develop, download, compile, debug, and run controllers in real-time while having full control over the laboratory environment including power control of the devices. They can monitor the laboratory with the audio and video tools that we utilize in our application. They can also collaborate with their colleagues using a shared whiteboard space and exchange ideas, also in real-time.

The first step in this development was to identify the major distance learning issues related to control engineering laboratory education. We have emphasized the importance of delivering practical experience to the students through active learning, total environment control, and collaboration issues covered with the application. The basis of this analysis was to identify what a local user does in the laboratory. It is extremely important that a remote user be able to do exactly the same operations. The main goal is to provide an environment which is the same as the local laboratory environment.

The remote user is provided with an interface to the laboratory. The graphical user interface is the first layer. It accepts user input and informs the client software about the request. The client talks to the server through the Internet. The server is interfaced to the laboratory equipment through a hardware/software interface. This interface allows the server to perform the user commands received.

The prominent feature of SBBT is that, unlike other distance learning applications, it provides a methodology to enable laboratory experiments involving

moving parts to be accessed remotely. This makes SBBT an ideal application for control engineering laboratories [4].

The next chapter outlines major laboratory education concepts as well as distance learning issues related to them. Chapter 3 outlines the laboratory environment control interface and gives an example of a typical session. Hardware configuration and in-lab communications are explained in Chapter 4. The experiences gained through the demonstrations and field trials are given in Chapter 5 along with a detailed description of the control engineering experiment used. Chapter 6 summarizes the application and gives pointers for open-ended research topics.

Related research is covered in detail in our previous work. The reader is referred to the following for further details: [5], [6], [7], [8], [9], [10], [11], [12], [13].

2. CONTROL ENGINEERING LABORATORY EDUCATION

A portion of this thesis is focused on outlining control engineering laboratory education concepts. Our motivation is to create a laboratory environment for the remote laboratory users that is functionally equivalent to the environment of the local users. The remote users should not lose anything from the practical experience gained through laboratory teaching. This led to our effort to replicate the similar environment with distance learning concepts [3]. A thorogh understanding of the educational issues is thus required.

Control theory and engineering concepts are put to practical operation in the laboratory. Students gain valuable insight and practice through observations. They learn through the collaborative environment provided in the laboratories as well as through the experimentation.

2.1. Distance Learning Concerns

The distance learning concept has been explained in greater detail in our previous work [3]. The following points summarize the main issues involved for remote laboratories.

- **activeness:** An application where the source of information is distant to the user should facilitate active participation of the user in the learning process. Active learning involves the user's active presence in the learning environment. This is traditionally given by video and audio transmission from the remote information source (e.g.. classroom, laboratory, etc.) [14], [15], [16], [17]. Another aspect of activeness involves providing the user with the tools to have complete control over the learning environment. Each "next step" in the learning process should be determined by the student, particularly allowing enough freedom for the user to make mistakes and learn from them.

One last issue in activeness criteria is the presence of collaboration during the learning process. An environment where the students can communicate with their peers or instructors must be provided.

- **data collection facilities:** When laboratory teaching is concerned, a very important issue in a remote operation is the existence of the tools to collect data from the laboratory environment [18], [19].

- **safety:** Whenever moving parts are concerned, safety issues must be addressed. Safety of the equipment from misuse and safety of local users from hazardous remote operation must be provided.

2.2. Traditional Control Engineering Laboratory Education

The laboratory teaching content and operations can be divided into three major areas: instrumentation, modeling and system identification, and implementation and verification.

2.2.1. Instrumentation

A thorough understanding of the experiment apparatus, and its operating principles and specifications is mandatory to proceed with laboratory operation. In this stage students form a conceptual and visual model of the overall experiment. They learn to power on and power down the system, identify inputs and outputs, and learn how the controller physically interacts with the controlled system. Specification documents are usually the main information source at this stage. It is also helpful if the students see a demonstration of the experiment before they start their own designs.

2.2.2. Modeling and Controller Design

A mathematical model of the system is defined either through extensive analysis or a system identification process. The controller is designed based on performance requirements. Students have to take data and perform certain measurements or be given the necessary information (e.g. specifications etc.) that is essential for them to proceed with this stage.

2.2.3. Implementation and Verification

Students are required to be physically present in the laboratory to implement the controller they have designed and to collect data to analyze the system performance. This stage provides the essential hands-on activity from which the students gain their practical experience.

2.3. Traditional Operations for Experimentation

The instrumentation, implementation and verification parts are the actual in-lab parts for traditional experimentation. The main operations performed in the laboratory by the local student are outlined as follows:

- **Powering on/off the equipment:** This is a general practice where the students need to activate the electrical machinery to start operation. This function also is used to reboot the PC, which is the controller device.

- **Developing the controller code:** Generally the control code is implemented in software on a controller device. We chose this device to be a PC with data acquisition boards. Students need to edit, compile,debug and test their controller code on this machine. The aim of the controller is to read sensor and/or stored data, compute the necessary controller action and then output the appropriate control command signals to the controlled device.

- **Running controller code and collecting data:** The control code needs to collect data from the system and be able to record these for further analysis. Control code may also need to read certain operational parameters (e.g. setpoints, controller parameters, timing information, etc.) from the storage device.

- **Resetting the system to a predefined state:** The equipment needs to be put into a state where it is ready to accept control commands.

2.4. Remote Laboratory Operations

A laboratory is a learning environment where students learn through their actions. Thus a successful remote laboratory application should allow the users to perform exactly the same operations as a local laboratory user might. This is needed for the remote users to gain the same experience as the local laboratory users. The basic operations that need to be performed by the remote user are derived from Section 2.2.

- Turn machines on/off, terminate action at any time.
- Develop controller code.
- Compile controller code.
- Run controller code.
- Create data files.
- Store collected data.
- Reset system to a state that will allow the system to take in controller commands.

The next two items are related to activeness concerns of distance learning issues.

- Visually observe the system.
- Communicate with others.

This list is the minimal functional set of operations for a remote control engineering laboratory. This functionality also ensures that the SBBT application covers traditional lab operations and all the distance learning concerns. They have to be addressed whenever the students are geographically distant to the actual information source which, in our case, is the laboratory.

All of these issues have been addressed in our application design through our five part interface [1], [2], [3]. The laboratory environment control interface provides all the necessary functionality to the remote students, gives them complete control of the remote environment and provides data collection facilities. Live video and audio from the lab provides the sense of presence in the laboratory. The collaboration tool (wb) allows the users participating in a session to exchange ideas. A panic stop button is provided for the remote user to intervene with the equipment operation in the laboratory whenever the need arises (e.g.. a fault occurring due to erroneous commands by the user or controller). The laboratory is also equipped with safety mats to protect the in-lab users from the equipment.

3. THE USER INTERFACE

The user interface is designed to address all the aspects of the laboratory distance learning concept. Our implementation choices reflect the methodology for delivering the open architecture we developed. The user interface has five parts (Figure 3.1): (1) the experiment, (2) laboratory environment control, (3) laboratory presence, (4) collaboration, and (5) safety.

3.1. The Control Engineering Experiment

This section refers to the apparatus being operated in the laboratory. The basic experiment is the controller device and the system being controlled. Not every experiment is a desirable candidate for remote operation even though our approach is as general as possible and could apply to almost any experiment. We outline the main characteristics of a good candidate in three major categories: economics, logistics, and presentability.

The cost of establishing a remotely accessible laboratory should be balanced with the cost of the experiments being used. It is desirable to share more expensive equipment since the cost of replicating it at different locations would be much more than making it available for remote access.

Logistically, the experiment should have the following characteristics:

- Remote power control: The experiment power is to be accessible remotely.
- Safety for people and property in the laboratory: Appropriate accommodations should be made in order to ensure the safety of equipment and in-lab users from hazardous remote operation.

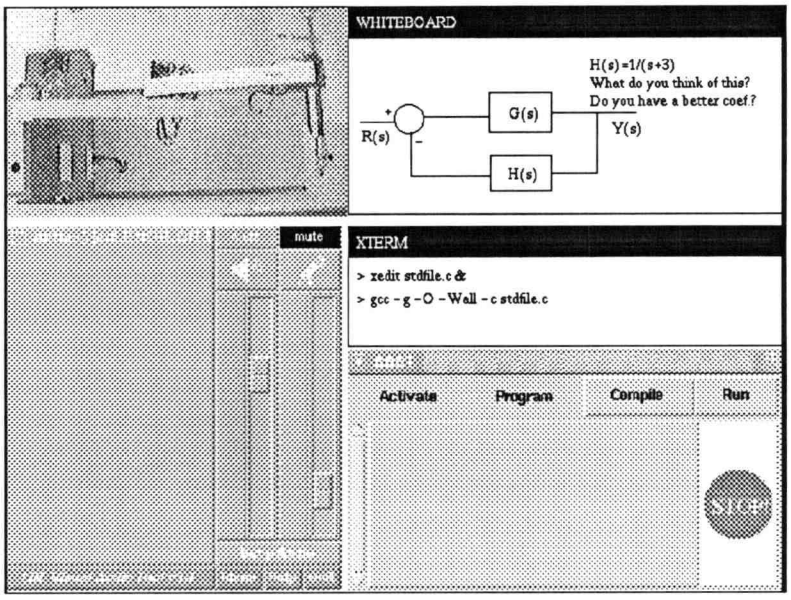

FIGURE 3.1. The Remote Lab User Interface for SBBT. This is the display the remote student will use to conduct experiments. Clockwise from the top left corner: 1) video window of the control experiment, a 3-DoF robot arm, 2)collaboration tool showing a block diagram and some discussion, 3) an Xterminal window representing the local development environment, 4) the laboratory environment control window with the panic stop button and 5) the audio configuration window.

- Ability to run without in-lab human intervention: The main idea is to provide flexible accessing mechanisms to the remote student. The laboratory equipment should be accessible 24 hours a day.

- A stable start state: The experiment being used should have a built-in stable predefined state to ensure the initial state of the system in any remote operation.

- At least one reset position: During remote operation, the user might want the apparatus to return to a predefined state. This may or may not be the same as the start state. The experiment being used should have these reset positions defined to guarantee these states to the remote user.

It is also mandatory to have a mechanism that will allow the remote student to download controller code and retrieve data from the laboratory equipment.

Appearance is a motivating factor for students. An experiment with visible moving parts is the best candidate for remote operation.

3.2. Lab Environment Control

This part allows the user to do all the operations a local user can perform in the laboratory (Section 2.2). The communication between the user and the laboratory is done over the Internet via a client/server interface. Table 3.1, [1], shows a list of commands that a user will issue.

3.3. Lab Presence

Laboratory presence is essential for a successful distance learning application. In our current implementation, communicating the laboratory presence feeling is done through audio and video broadcast from the laboratory. We chose to use the network tools *vic* [20] ,for video, and *vat* [21] ,for audio, developed at Lawrence Berkeley Laboratories (LBL). These tools provide a wide range of configuration options, including various coding schemes to broadcast to a variety of user platforms.

3.4. Collaboration Support Tools

It is important for the remote student to be able to share ideas with his peers or instructors while operating the environment. For this purpose, we investigated a tool developed at LBL called *wb* [22]. It consists of a shared piece of the CRT screen where the participating parties can communicate via writing or drawing. Other forms of communication could be e-mail or UNIX program "talk".

TABLE 3.1. Lab Environment Control.

Functions	**Explanation**
sbbt	Start up the application for a work session. If the experiment is already in use, a communication session is set up between the parties so scheduling can be negotiated.
quit	Release all SBBT resources.
stop	Immediate shutdown of motors.
reset	Put the experiment in predefined, stable state.
download	Transfer control code or data to the target controller.
reboot	Turn off power for several seconds to force the PC through a reboot sequence.
compile	Compile and link the control code on the target machine.
run	Execute the most recently compiled control code.
getdata	Transfer experiment output data to the user.

This part is also useful for handing over the control to a different user. The communication environment provides a "social protocol" to negotiate the equipment use.

3.5. Safety

Safety concerns are addressed in our application design in three different ways: (1) automatic hardware controls, (2) SBBT system control, and (3) student control. The laboratory is equipped with safety mats to ensure the safety of in-lab

users. The actuators of the moving parts are disabled if any person approaches dangerously close to the equipment. SBBT session has an internal watchdog that checks through periodic heartbeat signals whether network connectivity stays intact or not and thus the remote user is always there to intervene is a mishap occurs. If the network connection is severed for a long time (e.g. too many heartbeat signals missed) the equipment is shut off and the system is reset, disconnecting the user from the system. A notice is sent to the user at this point. As long as the network connection is sustained, the remote student has full control of the equipment power through the safety stop button that allows the user to shut off the equipment power whenever the need arises.

The next chapter covers the Laboratory Environment Control Interface that is basically the the piece that connects the user to the laboratory.

4. THE LABORATORY ENVIRONMENT CONTROL INTERFACE

The user can access the remote lab through the laboratory environment control interface (Figure 4.1) . This unit is the layer between the student and the underlying client software. The interface consists of a menu bar on top with clickable options, a window to display the desired information and a large clickable panic stop button.

The control interface is designed to operate with the minimal functional set. The commands cover all the actions a student might have performed if conducting the experiment locally. The control interface also enables users to view any information coming from the PC as if they were looking at its screen. The information that is reported by the client/server software is also displayed by this interface: the user is notified after each command has been received by the server. This window together with the audio and video tools are the most important tools enabling the remote student to have the same laboratory sense of interaction as a local student.

4.1. A Typical Session

The following is a list of actions constituting a possible SBBT session.

- Write the controller source code with a generic text editor. The source code must have the name stdfile.c and be stored as an ASCII file.
- Rebooting the PC (click on "Activate" and select "rebootpc") is recommended to guarantee its starting state.
- Download the controller code by clicking on "program" menu and selecting "download stdfile.c". After the download operation is complete, if the controller code requires a data file such as set point data, this can also be down-

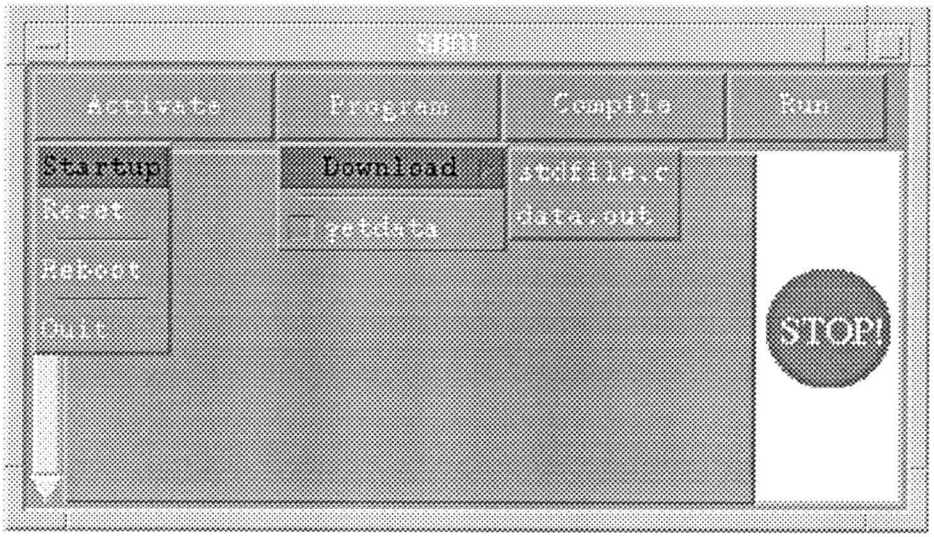

FIGURE 4.1. The Laboratory Environment Control Interface allows the user to issue commands easily.

loaded with a similar procedure (by selecting "download data.out"). For the current application the name of this data file has to be data.out.

- Compile the source code on the PC by clicking on "Compile". The compiler replies are displayed on the text window. If the source code is successfully compiled, go on to the next step.

- Reset the experiment by clicking on "Activate" and selecting "reset".

- Turn the motors on by clicking on "Activate" menu and selecting "startup".

- Run the controller code by clicking on "run".

- Halt and restart in the case of emergency, erratic behavior, or need to restart the system due to some reason, by clicking on "stop", "reboot PC" , "reset" and "startup" consequitively to restart the sequence. A brief pause is necessary after each command is issued in order to let the actions take effect.

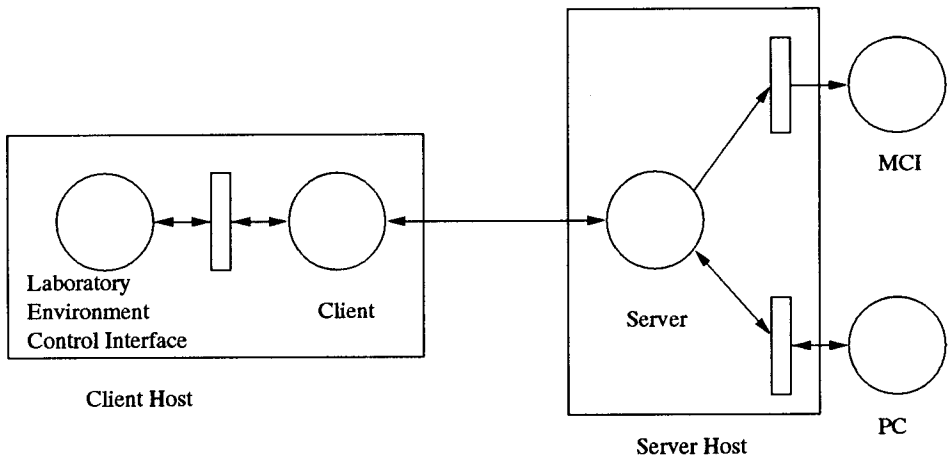

FIGURE 4.2. SBBT Software Connectivity.

- End the session by clicking "Activate" and selecting "quit". This actions exits the user interface and terminates the operation of the client.

4.2. Software Configuration

Major work on the software was done by Carisa Bohus and Dr. Lawrence Crowl. A very brief summary is given here and the interested reader is referred to [4], which gives a detailed account of the development process going from the iterative server which allows multiple parties to control the experiment, to the concurrent server, which basically shuts off all the users except one which has the control. In the latter scheme all the users are passive, given a passive user interface, but they still can observe the laboratory.

The main software configuration is outlined in Figure 4.2. The laboratory environment control interface is the piece that accepts the user commands. The client is a background process, which waits for a request from the user interface. Both pieces are resident on the remote user's host. The client-to-server communications is done over the Internet with UDP/IP. The server, which is a background

process on the server host, also waits for commands and executes them accordingly. The commands are implemented as two distinct pieces: one set communicates to the MCI box and another set to the PC.

5. HARDWARE CONFIGURATION DESIGN AND IN-LAB COMMUNICATIONS

The core part of a laboratory is the experiment itself, which, in a control engineering lab, consists of the controller and the controlled device. In the SBBT experiment implementation, a 386 PC is used as the controller device and a robot is used as the equipment being controlled. However, the SBBT application is general enough to be used with a variety of experiments [1], [2], [3].

Another layer providing the connectivity was added to enable remote access to the experiment. At the center of this layer is the server machine, which is a Sun SPARC5 UNIX workstation. The workstation resides in the laboratory and is linked to the lab equipment and the equipment power (Motor Control Interface (MCI)) (see Figure 5.1) using two dedicated RS232 serial lines. The workstation is also connected to the Internet through an ATM and an ethernet connection. It services the various possible user commands, and also acts as the multimedia server transmitting audio-video data from the laboratory and provides the shared whiteboard platform. The MCI provides the remote power control and supports hardware implementation of safety features. The hardware configuration supports full remote access to the lab and all the necessary safety features.

5.1. Motor Control Interface (MCI)

The safety features involved in a laboratory environment are handled through the custom-built MCI by basic power control. The commands are transmitted through one of the serial ports of the UNIX workstation. These commands are converted from RS232 to RS485 format in a intermediate conversion box. MCI takes in these commands through an RS485 line. Separate power controls are used for the controller device and the controlled device. The controlled device (robot)

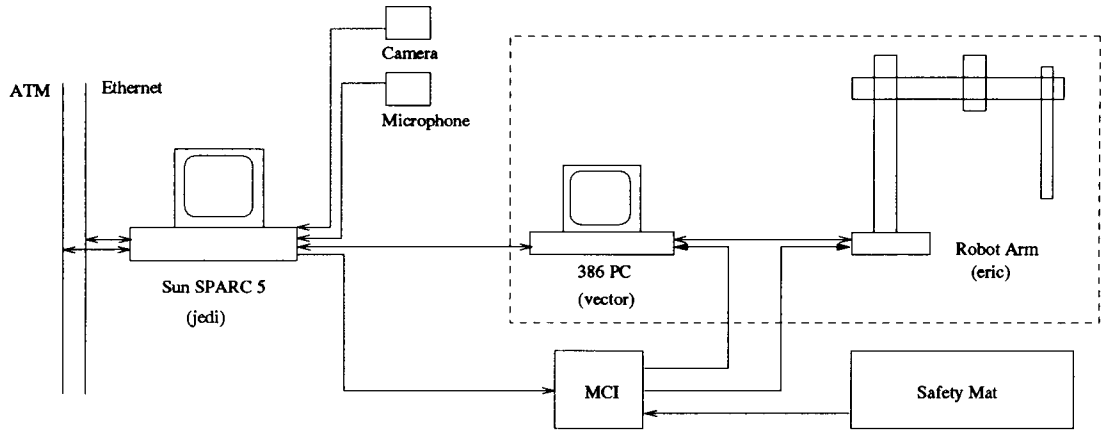

FIGURE 5.1. SBBT Hardware Configuration. The inset box shows the conventional hardware setup for a control engineering experiment. Outside this box, the workstation is used for remote connectivity and the custom-made Motor Control Interface box for power control.

power on/off are used by normal start and stop, by the panic stop button, by the safety mats, and by the system heartbeat. The controller (PC) power on/off is used to reboot the PC when it hangs up.

The MCI eliminates the need to have a person on-site to start up the experiment.

5.2. Workstation to PC Communications

This link is the key piece to the laboratory experiment. It allows the user to access the controller device through commands to the server. The source code and experiment data are communicated to the PC via this line, which is a full-duplex RS232 serial connection. Individual tasks assigned to the command structure are performed by shell scripts on the UNIX workstation and batch files on the PC. These instructions are thus implemented at the operating system level, giving the application flexibility and extended portability. The results of these communications and corresponding acknowledgments can be observed by the user through the audio and video tools and the data coming back from the PC.

5.3. Implementation of the Basic Functions

There are two sets of commands involved in remote supervision of the experiment. One set enables the communication to the MCI, and the other facilitates interactions with the controller device (PC). Each command sends an ASCII character string to the relevant serial port on the workstation. These ASCII character strings are stored in files and with the appropriate commands they are copied to the serial port. This structure makes it extremely easy to add new commands to the application.

5.3.1. MCI Commands

These commands require only one way (simplex) communications. The communication parameters are the serial port settings of 9600 baud, no parity, 8 bit characters, 1 stop bits. The ASCII strings and the functions are described below.

motoron : This command initiates the power on sequence. Before the power to the experiment is turned on there is an 8 second long alarm to caution the in-lab users. The command structure is the combination of the following two words : $1DO01 , $1DO00. These are sent sequentially through the serial line. The actual command is embedded in the last three bits of the command word. The last three bits operate in this sequence: first o o 1 , then o o o , which is the default (no commands) status. This combination sends a short pulse to the corresponding (least significant bit position in this case) command select switch of the MCI box.

motoroff : Turns the power to the motors off. The associated command string is $1DO02 , $1DO00. The last three bits operate in this sequence: first o 1 o , then o o o.

rebootpc : Turns the power to the PC off for 8 seconds and then back on. The word sequence is $1DO04 , $1DO00. The last three bits operate in this sequence: first 1 o o , then o o o.

5.3.2. PC Commands

Some commands require an application-level handshaking procedure and thus they require two way (full duplex) communications. The serial port settings are the same as the settings for the other port.

On the PC end, a TSR program enables the serial port (COM1:) inputs to be interpreted as keyboard inputs. With this architecture, the workstation acts like a student typing in desired commands on the PC console. The commands are described below.

reset : When invoked, it sends the appropriate executable file name to the PC. The file on the PC clears the controller board outputs (A/D registers) and thus terminates any control signal that might be going to the controlled device.

download : It initiates the controller source code download sequence. The source code transfer starts by the workstation informing the PC that a download is going to occur. The code is transferred after the PC is ready to accept the code.

compile : The compile routine is more involved. The routine is implemented as a batch file on the PC. After launching this batchfile the workstation waits for some form of data from the PC. The PC sends the compiler results to the workstation. On the workstation this data is captured as feedback to the user.

run : Runs the previously compiled program on the PC. Any standard output (screen output) is transferred to the workstation.

getdata : If the getdata option on the user interface is set, this command is invoked right after the run command to capture the data flow coming from the PC to transfer it to the user.

6. PROOF OF CONCEPT

This project has been demonstrated at various locations, and has been received with enthusiasm:

- March 26, 1995 : OSU Modern Communication Center opening.
- April 1995 : Portland State University.
- Spring 1995 Software Engineering Research Council IAB Meeting, Eugene, University of Oregon.
- October 31 - November 2 1995 : Educom 95, Portland OR, Convention Center.
- December 2 - December 8 1995 : Supercomputing 95, San Diego CA, Convention Center.

6.1. The Experiment

The experiment used for the demonstations is a 3-DoF robot arm with two joints and a plunger that moves vertically. The potentiometer sensors on the joints provide angular displacement data. The robot is connected to the PC, which is the controller device. The interface board in the PC is used both to provide inputs to the motor drivers on the robot and to sample the position data coming from the robot's joints. Each joint of the robot has a separate motor drive unit (three inputs and three outputs are needed).

The robot's motor drivers and the motion dynamics are highly nonlinear. Moreover, the effects of noise in the system are significant, making it difficult to design an effective controller.

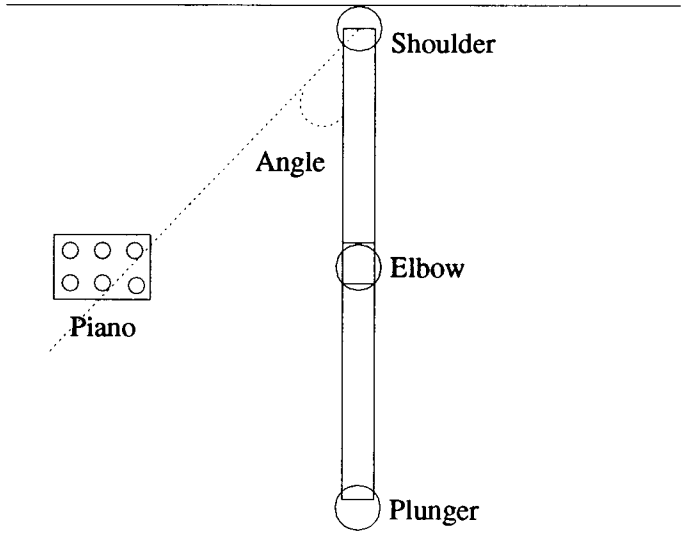

FIGURE 6.1. Top view of the robot arm

6.1.1. The Objective

The task chosen for the robot was to play a child's pop-up piano which has six keys, defined by their angular coordinates relative to the base of the robot. The robot has to go to each location and press the key on the keyboard, playing a tune. After pressing through the sequence of keys, the robot is to come back to the starting (reset) position.

6.1.2. Specifications

There are several requirements for the controller.

- The control signal cannot change too fast. Due to the weak coupling between the motor coils, a very fast signal change causes the motor to "slip" and the robot does not move even though the rotor spins.

- The resultant position error should be less than half the key size so the robot always hits the keys. Since the keys are of varying sizes, this error margin is defined by the smallest key size.

6.1.3. Observations

The physical system has certain properties that guide us while designing the controller.

- The shoulder joint on the robot is affected most by the nonlinearities, noise, and effects introduced by accelerating the heavier mass (the whole robot arm and the plunger) it carries.
- The plunger is least susceptible to noise and nonlinearities.

6.1.4. The Controller

Based on the observations and specifications, a separate controller has been designed for each joint on the robot arm. While closed loop control is employed for the shoulder and elbow joints, open loop control has been found to be adequate for the plunger.

System identification data from the robot was used to design these controllers. A linear time-invariant state space model of each joint was obtained. The order of the model for the shoulder was selected higher than that of the elbow's model because of its higher complexity. A third order model for the shoulder and a second order model for the elbow were found to be appropriate. These models are approximations derived from the original sytem identification data.

A PI (proportional-integrator) controller was designed for each joint based on its model. The whole control algorithm was based on the initiative to provide slow but accurate characteristics to the system dynamics. In order to slow the initial control peaks even further, a nonlinear filter was introduced to the control signal. This type of nonlinear modification can also be viewed as a form of gain scheduling. The control signal was attenuated for large deviations from the target (set point)

and preserved for small deviations. The nonlinearity used is a bell shaped curve corresponding to the formula:

$$f(\theta) = \frac{2e^{-|\alpha\theta|}}{1 + e^{-|\alpha\theta|}} \tag{6.1}$$

The performance of the controller is satisfactory, meaning the position error due to external disturbances and unmodeled nonlinearities in the system is less than the specifications, so no further modification on the control algorithm is needed.

6.1.5. Implementation

The controller is implemented in software on the PC. The pseudocode of this process is as follows:

```
get setpoints
start control
while job not finished(no more setpoints)
    begin
        read robot stat (position)
        calculate next control signal
        output control signal
        if on target perform keypress and goto next setpoint
    end
```

The controller samples the robot position as fast as possible and tries to react to it accordingly.

6.2. SBBT Application

It has been observed through the long-distance trials that the most important information for the remote user is the video component. By itself, this component

delivers almost all the feeling of laboratory presence necessary for comfortable user operation. Video provides instantaneous information about the consequences of user actions.

The quality of this type of feedback is directly related to the tool being used and the underlying technology that it utilizes. Multicast transmission reaches numerous hosts on the Internet at the same time but has a tendency to overload the network. Point-to-point operation reaches only one host but as observed in the demos, is the best solution for very long distance communications. At the San Diego point-to-point trial, the video performance was superior to the multicast video performance observed at Portland or Eugene (relatively closer locations to the source with similar bandwidth and frame rate settings as the former). Audio tools also have shown a similar behavior. Shared whiteboard does not have demanding bandwidth requirements. The video application has been found to require at least 200kbits/sec with 10frames/sec for satisfactory results. Video becomes quite choppy for lower bit rates, though it still provides valuable information and is acceptable, although not as appealing as a smoother vision.

The client/server does not have high bandwidth requirements since every command that requires fast response can fit into one UDP packet and the delay that the packet experiences is small - less then 1/3 seconds for round trip time. The real-time requirement is necessary only for the panic stop action.

7. CONCLUSION

We have successfully developed and demonstrated a distance learning application utilizing a new methodology to access remotely located engineering laboratories. Through remote laboratory environment control interface and multimedia tools, we are able to monitor and control laboratory equipment with moving parts. We have achieved full remote operation providing the remote users with the same sense of physical presence in the laboratory as the local users.

Videoconferencing tools have been clearly identified as the most important piece in this scheme. Their best usage has been identified. For shorter distances, a group/class work environment can be created using multicast transmission easily; but for much larger distances the most rewarding method is to use point-to-point transmission.

The application, which has high quality-of-service demands, helps us gain a new perspective on the existing network infrastructure and the communication tools. Such high bandwidth tools justify the need for high speed/high power networks. However, this still limits the availability of the resources, which now, with SBBT, has broken through spatial boundaries. We are able to access the laboratory from long distances over high-speed connections; Can students access the lab from their homes using modem lines? Can we have better quality of service? How can we guarantee this? There are various proposed ways of approaching these newly found problems. Further research on networking architectures, protocols or even infrastructure is a possibility. However, it is widely accepted that a more direct approach to the problem is to reduce the amount of data produced by the high bandwidth tools. Through employing video/audio compression data transmission can be reduced. Methods like subband coding for video/audio may reduce the adverse effects

of packet loss. However, these processing schemes can be computationally very intensive. This brings us to the question "Is there a trade off between CPU power and network speed?" In high speed networks, most of the time the bottlenecks, as we observed in our demonstrations, are the processing nodes (the source and the receiver).

Approaching the problem from the networking side reveals a whole set of new possibilities. IP multicast, scheduling policies, rate control on sources, and congestion control are open ended research topics. Enforcing quality of service requirements on the network and being able to guarantee this for the duration of communication is important. Modeling the network, identifying the network states and trying to utilize feedback data may help to obtain a globally optimal traffic control scheme. The existence of such a scheme is, of course, a question itself.

Revealing important issues involved in multimedia applications over communication networks, SBBT allows us to have a starting point and a concrete application to be used with future research. SBBT is a milestone in our efforts to create a more advanced, widely accessible, highly flexible working and learning environment for our students.

BIBLIOGRAPHY

[1] Carisa A. Bohus, Burçin Aktan, Molly H. Shor, and Lawrence Crowl, "Running control engineering experiments over the internet", To appear in *IFAC World Congress*, San Francisco, June 1996.

[2] Carisa A. Bohus, Burçin Aktan, Molly H. Shor, and Lawrence Crowl, "Running control engineering experiments over the internet", *Oregon State University, Computer Science Department Technical Report*, vol. TR-95-60-7, August 1995.

[3] Burçin Aktan, Carisa A. Bohus, Molly H. Shor, and Lawrence Crowl, "Distance Learning Applied to Control Engineering Laboratories", Manuscript to appear in *IEEE Transaction on Education.*

[4] Carisa A. Bohus, "Implementing Remote Laboratories for Control Engineering: Foundations for Distance Learning", MS Thesis, Department of Computer Science, Oregon State University, Mar. 15, 1996.

[5] Nadine E. Miner and Sharon A. Stansfield, "An interactive virtual reality simulation system for robot control and operator training", in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1994, vol. 2, pp. 1428–1435.

[6] Sean Graves, Larry Ciscon, and J.D. Wise, "A modular software system for distributed telerobotics", in *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Nice, France, May 1992, pp. 2783–2785.

[7] George V. Kondraske, Richard A. Volz, Don H. Johnson, Delbert Tesar, Jeffrey C. Trinkle, and Charles R. Price, "Network-based infrastructure for distributed remote operations and robotics research", *IEEE Transactions on Robotics and Automation*, vol. 9, no. 5, pp. 702–704, October 1993.

[8] Sukhan Lee and Hahk Sung Lee, "Modeling, design, and evaluation of advanced teleoperator control systems with short time delay", *IEEE Transactions on Robotics and Automation*, vol. 9, no. 5, pp. 607–623, October 1993.

[9] Steven Gentner, Nick Rothenberg, Carl Sutter, and Jeff Wiegley, "The mercury project - robotic tele-excavation", http://www.usc.edu/dept/raiders/, 1 Sep 1994 - 31 March 1995.

[10] George Bekey, Steven Gentner, Rosemary Morris, Carl Sutter, and Jeff Wiegley, "The tele-garden", http://www.usc.edu/dept/garden/, 1995.

[11] Active Robotics Group, "A new technology initiative", http://skynet.reading.ac.uk, August 1995.

[12] Announcement in *Currents*, a publication of the Electrical and Computer Engineering Department, Carnegie Mellon University, Spring 1995.

[13] Mike Driscoll, ", from conversations, GPIB bus project, Spring 1995.

[14] Roger C.Schank, "Learning via multimedia computers", *Communications of the ACM*, vol. 36, no. 5, pp. 54–56, May 1993.

[15] James A. Schnepf, David H. C. Du, E. Russel Ritenour, and Aaron J. Fahrmann, "Building future medical education environments over atm networks", *Communications of the ACM*, vol. 38, no. 2, pp. 54–69, February 1995.

[16] Chungming An, Brian T. Barcelo, Joyce A. Inkrott, Arthur R. Snowdon, and Kenneth J. Trojniar, "A multimedia distance learning trial using isdn bri", *ATT Technical Journal*, pp. 15–21, January/February 1993.

[17] Beverly Park Woolf and Wendy Hall, "A multimedia pedagogues: Interactive systems for teaching and learning", *IEEE Computer*, pp. 74–80, May 1995.

[18] Denis Newman, "School networks: Delivery or access", *Communications of the ACM*, vol. 36, no. 5, pp. 49–51, May 1993.

[19] Dick Ruopp and Shahaf Gal, "The labnetwork", *Communications of the ACM*, vol. 36, no. 5, pp. 35–36, May 1993.

[20] Steve McCanne and Van Jacobson, "vic (VIC 2.6 BETA)", Available by ftp from ftp.ee.lbl.gov under conferencing/vic (e-mail contact mccanne@ee.lbl.gov), 1994.

[21] Van Jacobson and Steve McCanne, "vat (Beta release)", Available by ftp from ftp.ee.lbl.gov under conferencing/vat (e-mail contact van@ee.lbl.gov), 1994.

[22] Van Jacobson and Steve McCanne, "wb (LBL whiteboard) version 1.59, Beta release", Available by ftp from ftp.ee.lbl.gov under conferencing/wb (e-mail contact van@ee.lbl.gov), 1994.